CHAPTER

10

CREATING IMAGE ROLLOVER EFFECTS

Using JavaScript to Control Image Display

In this chapter, you will:

- Learn about rollover effects
- Create rollover effects with ImageReady
- Create rollover effects with JavaScript
- Optimize rollover effects with JavaScript

Anyone who surfs the Web is familiar with rollover effects—one image swaps with another when you point to an object with the mouse. These effects are sometimes also referred to as image swaps or mouseovers, though in this chapter they are simply called rollovers. Almost every professional Web site uses rollovers in some way. Rollovers can be functional, and provide the user with extra information when the mouse is rolled over a link. Rollovers are usually decorative, however, and reinforce that the mouse is pointing to a link.

The most basic rollover is one where moving the mouse pointer over an image replaces the current image with another. You can enhance the rollover to include a third image for when the user actually clicks the current image. Another type of rollover swaps a secondary image with a separate image when the user points to the image that initiates the rollover effect.

You can create rollover effects by hand-coding the necessary JavaScript, or by using software such as ImageReady, Fireworks, or many other available programs. These programs create the JavaScript for you, making the process easier. However, relying on the code generated by these tools limits what you can do. If you learn the basics of JavaScript, you can edit the code generated by these programs to create any effect you want.

Whether functional or decorative, mouseovers can add professional polish to a Web page. In this chapter, you learn to create simple and complex rollover effects using JavaScript, coding by hand, and using ImageReady.

Understanding Rollover Effects

In addition to creating graphic files, you can use ImageReady to generate the HTML and JavaScript for creating basic rollover effects.

Creating rollovers with ImageReady is very similar to creating animations with ImageReady. Instead of creating multiple animation frames, you create multiple rollover states. Each rollover state you create results in a separate image file, which is associated with a specific position of the user's mouse relative to a hyperlink. The hyperlink is usually an image, but does not have to be. Possible rollover states include: Over, when the pointer is over the link on the screen, Down, when the user actually clicks down on the image, and a few others, which are covered in later sections.

Instead of playing in sequence, the different states appear only when the user initiates them, either by rolling over or clicking an image. Like animation frames, rollover states in ImageReady are composed of one or more layers with various settings for opacity, position, and style.

Each rollover state requires its own image. Using several rollover states with a link requires that the user load additional graphics files.

Unlike creating animations, you can create rollover effects without any special software by creating the images for each rollover state, and manually coding the necessary JavaScript and HTML. For complex rollover effects, such as having a rollover effect swap an image elsewhere on the page, you must produce the code manually. For most rollover effects, however, you will find it easier to use ImageReady.

A key advantage to creating rollovers with ImageReady is that you create images for all rollover states at the same time, using the same layers. This ensures that the rollover effects perfectly match across different states.

Before you start creating rollover effects, you need a better understanding of them.

Designing a Basic Rollover

Rollovers are commonly used to show the selected button in a navigation bar. In this usage the rollover image is usually a highlighted version of the initial image, as in Figure 10-1.

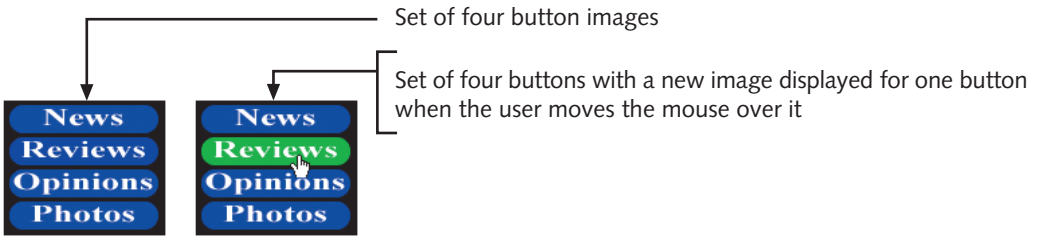

Figure 10-1 An image and its rollover version

The highlighted version serves only to indicate that the user's pointer is over a link. This is usually just a decorative effect, but can also let users know which link they are about to click. If you use many small buttons near each other, you may want to use rollover effects to make the navigation easier.

Some designers use a blurred image as the initial image, requiring the user to point to it to see the in-focus version, as shown in Figure 10-2.

Figure 10-2 A rollover effect where only the rollover version is clear

Avoid the blurred image technique because it hinders the usability of the site more than it helps. It forces users to search for the identity of the links, rather than making the identity obvious. Instead of making the navigation easier, using rollover effects this way actually makes the navigation more difficult.

A more elaborate type of rollover effect includes three images: one image in its normal state, a different graphic when the pointer is over a link, and a third graphic for when the user actually clicks down on the link. This sort of rollover typically uses a button image for the initial state and a more brightly colored or highlighted version of the same button for the Over state. For the Down state, the Over button can be inverted or rotated 180 degrees. This causes the 3-D effects to invert, as shown in Figure 10-3, enhancing the effect of the button being pushed down.

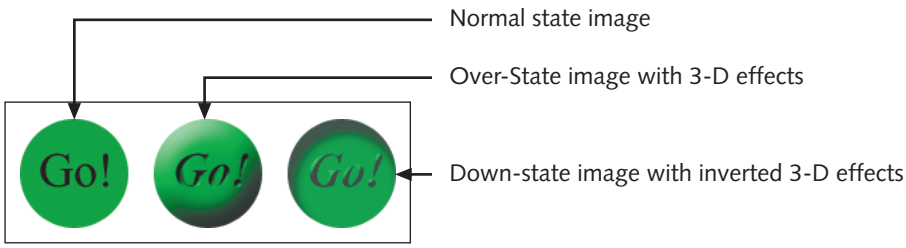

Figure 10-3 Three states of a rollover

The benefits of using a three-state rollover effect are that the buttons seem more realistic and more similar to buttons on physical devices. Using the Down state also can help reinforce the idea that the user successfully clicked the link.

While using three images makes using a button seem more realistic, it also forces the user to download more image files, and can increase the download time. On the other hand, rollover images are usually used in navigation, so the same images are used on every page and need to be downloaded only on the first visit to a site, after which point they are cached. When deciding to use rollover effects, make sure that the benefits of using rollovers outweigh the drawback of the extra download time. While a few graphical rollovers help make a site look more professional, too many make the site look amateurish.

Image swaps that use multiple images can be used for instructional purposes. Mousing over a link or image can cause an informational text image to appear. For example, an educational resource site might have images of human anatomy where mousing over specific parts of the image causes the definition of that part to appear in a separate area on the page.

Like most other types of Web graphics, image rollovers can be overused. Therefore, you should use rollovers only when they provide a useful function for the Web page.

Using the Rollover Palette in ImageReady

When you create rollovers in ImageReady, you work primarily with the Rollover palette and the Layers palette. The Rollover palette is found next to the Animation palette, as shown in Figure10-4.

When you create or open an image, it appears in the Rollover palette in the Normal state—the initial image as it appears before the user points to it. When you have more than one state for a rollover, you can select only one state at a time. The selected state has a black outline around its thumbnail preview in the Rollover palette, and appears in the document window. Each state is treated as a separate image, using one or more layers in the Layers palette.

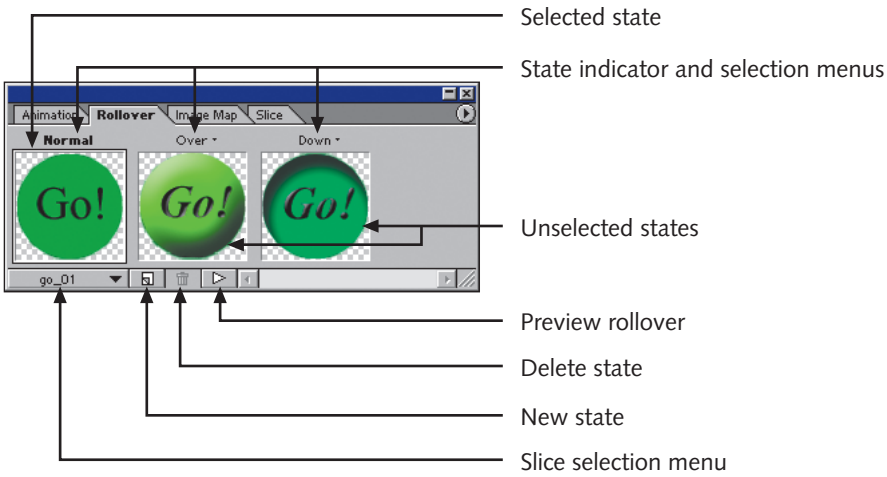

Figure 10-4 The Rollover palette in ImageReady

The Rollover palette includes buttons at the bottom of the palette. These include:

- **Slice selection bar**. You can use slices to make complex layouts of images using HTML tables. This book covers slices in detail in the Creating Sliced Images chapter.
- **New state button**. This adds a new state to the rollover. The first state is always the Normal state. Each new state takes the layer properties of the preceding state, so the second state has the same layer properties as the initial image. You can then use the Layers palette to make changes to the image in the new state.
- **Trash**. Delete rollover states as you delete animation frames. Select the state, then click the Trash button in the Rollover palette, drag the state to the Trash button, or select Delete State in the Rollover palette menu.
- **Preview**. Click this button to preview the rollover in ImageReady. Although it resembles a Play button, clicking Preview does not play a sequence of states. It allows you to roll over and click the Image window to view the rollover effect.

You also can preview the rollover in ImageReady by clicking the Rollover Preview button in the Toolbox, shown in Figure 10-5.

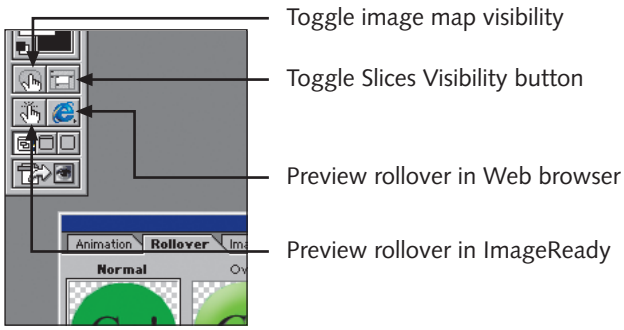

Figure 10-5 Preview buttons

You should preview the rollover in a browser to make sure it functions as you intended. Even after you have successfully viewed the rollover on your computer, you should view it on other systems as well. Rollover graphics might appear as you intended on one computer, but not appear at all on another. Rollover effects rely on simple JavaScript programming. A browser that does not support JavaScript or that has JavaScript capabilities disabled cannot display rollover effects.

Creating Rollover States

In ImageReady you can create rollovers from one layer or a set of layers. Each state uses some or all of the layers. There is one state for each type of mouse action. The available states are:

- *Normal*: This is the initial image displayed in the Web page.
- *Over*: The image displayed when the mouse rolls over the link. This is similar to using the Hover attribute in CSS.
- *Out*: The image displayed when the mouse rolls off the link. Normally this will be the same image as the Normal state.
- *Down*: The image displayed when the mouse button is pressed over the image. This is similar to using the Active attribute in CSS, or the ALINK attribute in HTML.
- *Up*: The image displayed when the mouse button is released. This is usually the same image as the Over state. This is similar to using the Visited attribute in CSS, or the VLINK attribute in HTML.
- *Click*: This is like the Down state, except the state reverts to Normal when another state is initiated.
- *Custom*: This state is reserved for uses that don't apply to creating rollover states.
- *None*: This creates an image that will be preloaded but will not appear in the Web page, so you do not need this state for Web graphics.

You probably will work with the Normal, Over, and Out states most often. The Down, Up, and Click states are used when you have separate images appear at the moment the user clicks a link. In the time it takes the user to click a link, the browser will display the image for the Click state or the images for the Down and Up state. These images appear so quickly that they are seldom worth the extra download time.

You cannot create a customized state, because JavaScript can deal only with the existing states. A state of None means the image will not be displayed at any time as part of a rollover effect.

Create rollover states in ImageReady:

1. Open **fox.tif** from the Data Disk.
2. Show the Rollover palette by selecting **Show Rollover** from the Window menu. You will see a thumbnail of the new image.
3. Click the **New state** button at the bottom of the Rollover palette. A second state of the Over type is added to the palette.
4. Click the word **Over** to display a menu of options listing all the available states. Select **Down**.
5. Leave this file open.

During rollovers, images are swapped depending on the state of the mouse and browser. Entire images are replaced, so you do not need to set the frame disposal method or the delay as you do with animations.

Creating Rollover Effects with ImageReady

Creating and editing rollover states is similar to creating and editing animation frames. Each state is composed of one or more layers, as indicated by the visibility icons next to the layers in the Layers palette. You can select only one state and one layer at a time.

Using Layers to Create Rollover Effects

As with animations, layer changes such as opacity, position, or style affect only the selected state. Other changes, such as using filters or brush tools, affect all states that display the layer.

To create rollover effects by changing opacity:

1. With fox.tif open, select the **first frame**.
2. In the Layers palette, set the opacity to **50**%.
3. Preview the rollover effect in ImageReady by clicking the **Preview** button in the Rollover palette or the **Rollover Preview** button in the toolbox.

4. Click the **image**. The image should be faint until you click it, at which time it appears brighter.
5. Preview the animation in a browser. You should see the same effect. Note the JavaScript code used to create the rollover effect. This will be explained later in this chapter.
6. Save this image by setting an appropriate optimization setting in the Optimize palette, then selecting **Save Optimized As** from the File menu. You will be asked to save an HTML file named fox.html. This file contains the HTML and JavaScript necessary to display the rollover images. Save this file in a folder named **chapter10-1**. ImageReady will automatically create a folder named images that contains the actual graphic files. The details of saving rollover files in ImageReady are covered later in this chapter.

Layer changes involving opacity and position affect only the selected state. You can apply changes to all states by selecting the layer, and then selecting Match Layer Across States in the Rollover palette menu.

Because layer edits using filters or tools from the toolbox affect all states that display that layer, you often need to create multiple layers for an image in order to use these tools.

To create rollover effects with multiple layers:

1. With fox.tif open, select the **layer** in the Layers palette.
2. Select **Duplicate Layer** from the Layers palette menu.
3. With the new layer selected, use the Hue/Saturation dialog box and boost saturation to **100**%.
4. In the Layers palette, set the opacity to **100**%.
5. Create a new state in the Rollover palette. Set the state to **Over**.
6. Select the **Normal** state and make the new layer invisible.
7. You should now have three states: Normal, Down, and Over. The order in which the states appear in the Rollover palette does not matter. Select each state in turn and make sure the proper layers are displayed for each state. The Rollover and Layers palettes should look like those in Figure 10-6. The Normal state should display only Layer 1 at 50% opacity. The Down state also should display only Layer 1, but at 100% opacity. The Over state should display the new layer, Layer 1 Copy, at 100% opacity. It does not matter if this state displays both layers or only the top one because Layer 1 Copy completely occludes Layer 1.
8. Preview the rollover in a browser. The image should get much brighter when you mouse over it, and somewhat dimmer when you click it.
9. Save the HTML and images in a new folder named **chapter10-2.** Close **fox.tif**.

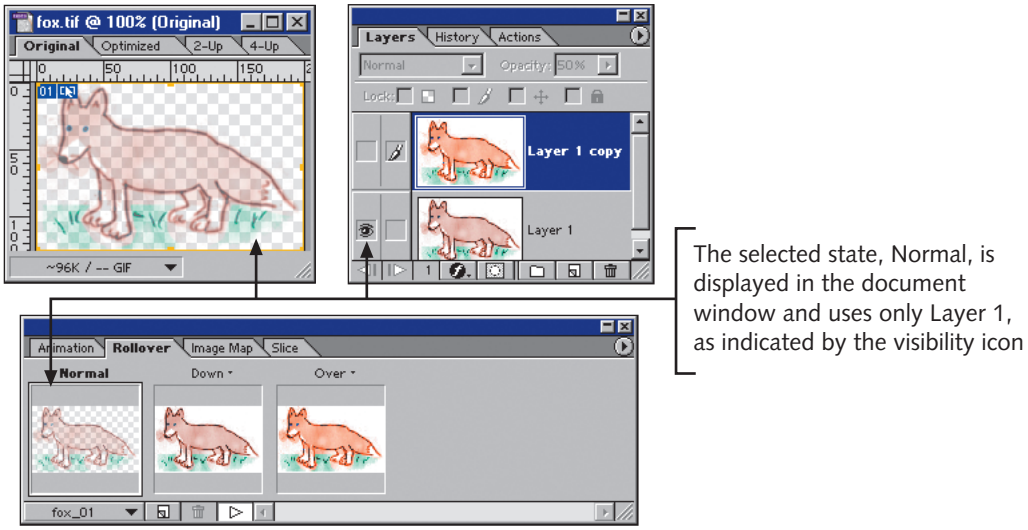

Figure 10-6 The Rollover palette with three states displaying different layers

You can copy and paste rollover states just as you can copy and paste animation frames. In fact, you can copy rollover states and paste them into animations, and vice versa. When pasting, make sure to use the Paste Rollover State command from the Rollover palette menu. Selecting Paste from the Edit menu will paste the contents of the Clipboard into the current layer. When you paste a state, the existing state is replaced with the layer visibility options of the copied state.

Using Styles to Create Rollover Effects

One advantage to using ImageReady for creating rollovers is the convenience of using styles to create the rollover states. A style is an effect or combination of effects applied to a layer. In previous chapters, you applied layer effects such as Drop Shadow or Bevel and Emboss. ImageReady offers many preset combinations of multiple layer effects in the Styles palette, shown in Figure 10-7. For example, the Meshed Gradients style is a combination of the Outer Glow, Inner Glow, Bevel and Emboss, and Gradient Overlay layer effects.

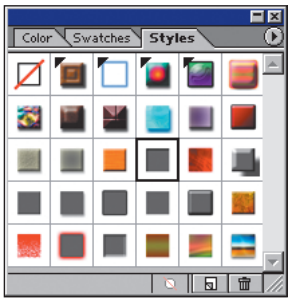

Figure 10-7 The Styles palette in ImageReady

You can edit a style by clicking it in the Layers palette. The Layer Options palette then displays the style options. Figure 10-8 shows the Styles, Layer Options, and Layers palettes when editing a style.

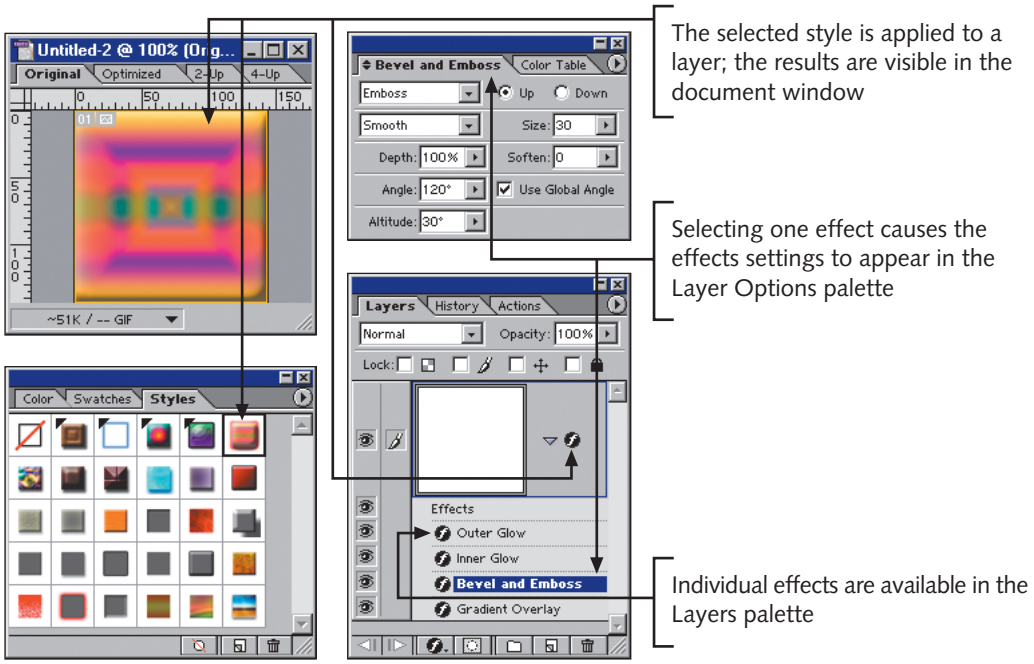

Figure 10-8 Editing a style

The selected style is applied to a layer and the results are visible in the document window. Specific effects are available in the Layers palette, where they can be enabled or disabled individually by clicking the visibility icons. Selecting an effect causes the Effects settings to appear in the Layer Options palette. There, you can edit the Effects settings.

To apply a style to a layer:

1. Create a **100 × 100-pixel image** with a **transparent** background.
2. Show the Styles palette by selecting **Show Styles** from the Window menu.
3. Locate and select the style named **Button-Up**. You should not see any changes to the image in the document window. Layer styles need image data to have any effect.
4. Select the **Paintbrush** tool and set the foreground color to **black**.
5. Draw a **star** in the document window. Instead of a normal black line, the drawn line will have a beveled, 3-D effect, as shown in Figure 10-9.

6. In the Layers palette, click the right-pointing **triangle** next to the effects icon in Layer 1. This expands the list of layer effects used in the style.
7. Hide the Drop Shadow effect by deselecting the **visibility icon** for the Drop Shadow effect in the Layers palette.
8. Save the optimized image as **style.jpg**.

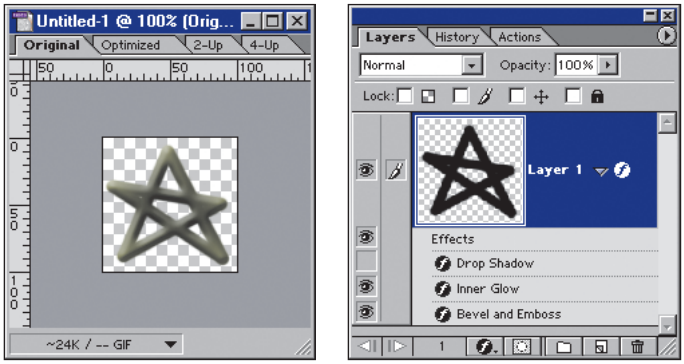

Figure 10-9 An image with applied style

Some styles not only combine multiple layer effects but also automatically create multiple rollover states. When you select a style, you automatically create a full set of rollover images. This is a quick way to get immediate results. For most professional jobs, however, you need to control the images yourself, and you will not use styles.

Rollover styles are indicated by a black triangle in the upper-left corner of the style thumbnail, as shown in Figure 10-10.

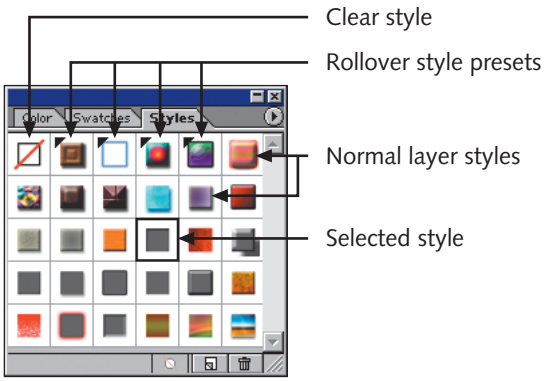

Figure 10-10 Layer rollover styles

Some styles generate many states, and others just one or two. You can add more states if you want, and edit the layers to create additional effects.

Create a rollover in ImageReady using styles:

1. Create a new image in ImageReady by clicking **New** on the File menu. Set the size to **72 × 72 pixels** and make the background **transparent**.
2. Fill the layer with **white** using the Paint Bucket tool.
3. Click the **Wood 3-state Rollover** style in the Styles palette.

   In the Rollover palette you see the three states for this style of rollover, as shown in Figure 10-11.
4. Click the **Preview in Default Browser** button. The three rollover images and all necessary HTML and JavaScript are generated and displayed in your browser.
5. Roll over and click the **image** in the browser to see the effect.
6. Save this optimized set of rollover images with an HTML file named **wood.html** in a new folder named **chapter10-3**.

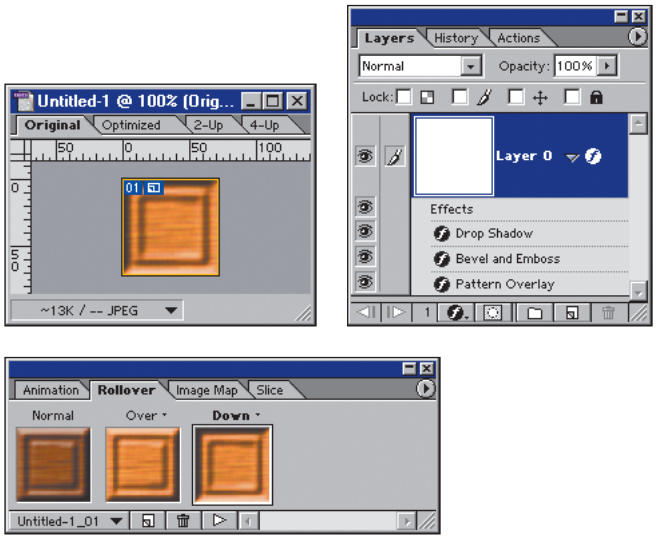

Figure 10-11 Creating a rollover effect using a rollover style

In addition to the default rollover styles in the Styles palette, you can load additional styles from the Styles palette menu.

Using Animation in Rollover Effects

When designing rollover graphics, you normally use a plain image for the initial state, and a highlighted image for the Over state. One way to highlight an image is to animate it. To animate a state in a rollover, select the state you want to animate in the

Rollover palette, select the Animation palette, and then create the animation. For more information on animating images, see the chapter on Creating Animated Graphics. You can create an animation for every state in a rollover, but this can lead to large files. Whereas one static image with no rollover effects might be 10 KB, a rollover effect with a similar design but three animated states might be 60 KB.

Create an animated rollover:

1. Open **fox.tif** from the Data Disk.
2. In the Rollover palette, create a new **Over** state.
3. With the Over state selected, show the Animation palette.
4. In the Animation palette, create a **second frame**.
5. With the second frame selected, use the Move tool to move the contents of Layer 1 in the Layers palette. Move the layer **10 pixels** to the right.
6. In the Animation palette, set both frames to delay for **0.2** seconds and set the looping to **Forever**.
7. Preview the file in a Web browser. You should see a static version of the fox. When you move your mouse over the image, the fox should move back and forth.
8. Save the optimized file as **fox.html** in a new folder named **chapter10-4**. This creates two images—the second image is animated.

It can be confusing to generate rollover states and animations at the same time from the same set of layers. You may find it is easiest to set up your layers for the rollover states, and edit the rollover state images first. Then use the same set of layers to animate individual states.

Saving HTML Files in ImageReady

When you save an image with rollover states, each state is saved as a separate image file. The names of the images are based on the name of the slice used in the image. **Slices** are sections of images, and are covered in detail in the Creating Sliced Images chapter. In this chapter, all rollovers use one slice that fills the entire image. Sometimes you will see outlines over the image with a blue box containing the number 01, as shown in Figure 10-12. This number displays the slice information about the image. You can hide the slice information by deselecting the Toggle Slices Visibility button in the Toolbox, shown in Figure 10-5.

Figure 10-12 Slice information about an image

The JavaScript generated by ImageReady must be able to refer to separate rollover images by name. By default, the image names are based on the name of the original image, plus an extension describing the rollover state. For example, if you create an image and build three rollover states from it, the images might be named Untitled_1.jpg, Untitled_1-over.jpg, and Untitled_1-down.jpg.

You can give the images other names if you wish. This will not affect the rollover effects, but may make managing your image files easier. When your image is open in ImageReady, you can name it using the Slice palette, shown in Figure 10-13. Show the Slice palette by selecting Show Slice from the Window menu.

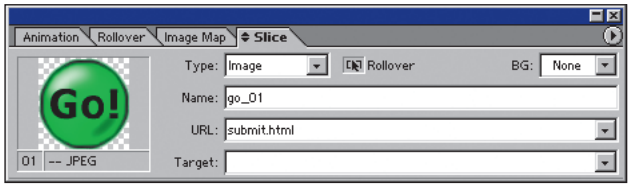

Figure 10-13 The Slice palette in ImageReady

In the Name field you can name the slice. This name then determines the names of the images generated as part of the rollover. In the URL field you can enter the address of the page to which the rollover should link. You also can edit this information directly in the HTML file after it is generated.

Save rollover effects by selecting Save Optimized As from the File menu. Make sure you have selected appropriate optimization settings for the rollover images. Unlike animation files, rollover graphics can be saved as GIF, JPEG, or PNG. In the Save Optimized As dialog box, you have the option of saving images only, HTML only, or image and HTML files both. Make your selection in the Save as type field in the dialog box. Choose HTML and Images (*.html). In the File name field, choose a name for the HTML file.

Once you save the rollover images and have the HTML file, you can make multi-image rollovers more sophisticated by adding images and changing and adding names in the HTML code. You often will have to manually edit the JavaScript generated by ImageReady, because the software cannot create every effect you might need. To do so, you need to understand how to create rollover effects manually with JavaScript.

Creating Rollover Effects with JavaScript

Until very recently all client-side effects, including rollovers, had to be coded by hand in JavaScript. Now many WYSIWYG HTML editors and some image editors, such as ImageReady, write JavaScript code for rollovers that you can use in Web pages. To create certain complex rollover effects, however, you must modify this code or create new JavaScript code yourself.

You do not need to understand every aspect of JavaScript to implement JavaScript rollovers. However, a lack of understanding might result in rollovers that work only on some browsers. Most browsers support JavaScript rollovers, but the oldest browsers cannot display any rollovers at all. Netscape version 4 and higher, and Internet Explorer version 4 and higher both display JavaScript rollovers.

Before continuing, locate the file named fox.html that you created in the last section, or find fox.html in the Data Disk. Open the file in a text or HTML editor so that you can see the code. The code will be explained in detail in this section.

Working with JavaScript

HTML is a markup language—you use it to control the display of text and graphics in a Web page. JavaScript is a programming language—you use it to write scripts that take inputs and produce corresponding output. Every program in any language must be run, or **executed**, in a specific environment for it to work. A program written in C can be compiled into a standalone application that runs on your desktop. A droplet created in Photoshop or ImageReady is similar, and also runs on your desktop. A CGI script written in Perl runs on a Web server. A Java applet runs within the browser in an application called a Java Virtual Machine (JVM) that works like a plug-in. JavaScript runs directly in the browser, needing no plug-in.

Java and JavaScript have a few similarities, but are otherwise unrelated. Both are based on the C programming language, and both are used primarily for Web-based applications. Java, however, is a complete, complex language which can be difficult to learn, but can be used to create server-side or desktop applications in addition to the applets used in Web pages. JavaScript is not as robust as Java and can be used only in Web pages. However, it is easy to learn and use.

JavaScript manipulates elements of the browser window and pieces of Web pages. For example, you can use JavaScript to print the current date to the screen or to control the size of the browser window. JavaScript is written directly into a Web page, so you can view and copy the source as you would with HTML or CSS. Like CSS, which is not a programming language, JavaScript can be written using any of three conventions:

- **inline**, meaning the code is placed within an HTML tag
- using **SCRIPT** tags elsewhere in the same HTML document
- in an **external file** that can be accessed by any other HTML file

Using Inline JavaScript

An example of inline JavaScript is:

```
<BODY BGCOLOR=#FFFFFF ONLOAD="preloadImages();">
```

The BODY tag takes a JavaScript attribute called ONLOAD, which takes the value preloadImages(). ONLOAD describes the state of a Web browser when all elements of a Web page have been downloaded. The onMouseOver attribute, which you use in rollovers, describes the state of a Web browser when the mouse pointer is directly over a particular link. You can see the onMouseOver and onMouseOut attributes within the A tag near the bottom of fox.html. These attributes are flags for the browser.

Essentially, the example above tells the browser "when all elements have been downloaded, execute the following script, named preloadImages()". In the example, preloadImages() is a **function**, which is a list of instructions. A function is a self-contained sequence of commands that can be called by other functions or called from an attribute such as onMouseOver. Some functions are included with the JavaScript interpreter that is part of the Web browser. You also can write your own functions.

Using the SCRIPT Tag

The following JavaScript example is not inline, and instead relies on the SCRIPT tag in HTML.

One of the simplest JavaScript codes displays text on the screen. The traditional first program in any programming language writes the message Hello World! to the screen:

```
<script>
document.write("Hello World!");
</script>
```

The above script can be placed anywhere in a Web page, and displays the text on the screen.

Here, write() is a JavaScript function included in the standard JavaScript function library. It takes the value specified in parentheses and displays it in the document, which is another term for the Web page. The JavaScript code in the preceding example writes Hello World! in the current document.

This script requires the use of the SCRIPT HTML tag. Most tags instruct the browser how to display the text that follows the tag. The SCRIPT tag, however, tells the browser that what follows is an executable command.

In fox.html, SCRIPT tags are used to contain three functions. These functions call each other, and are called by the onMouseOver and onMouseOut attributes.

In JavaScript, spaces, tabs, and carriage returns are all treated as equivalent white space. It does not matter if you use one space, ten spaces, a few tabs, or several carriage returns; they are all treated as a single space. You can format JavaScript using tabs, spaces, and carriage

returns as you would in HTML, though you need to indicate the end of a line in JavaScript with a semicolon (;). Every distinct line of JavaScript code must end with a semicolon.

Using JavaScript in an External File

An external JavaScript might look like the following, and would be placed in a separate file. You can choose any name you want for the file. The following code is the same as the previous example.

```
document.write("Hello World!");
```

This JavaScript does not need the SCRIPT tag around it. However, to access the script in an external file, you must use the SCRIPT tag, indicating the filename. If the script above were saved in a file named image.js, you would access it using the following code:

```
<script src="image.js"></script>
```

You can write the simplest JavaScript inline. If you know you will use the same function in many places, you should use an external file.

Creating Simple JavaScript Effects

Using JavaScript to write text on the screen is not useful, because you can simply write the text directly in HTML without using JavaScript. If you want to do something more dynamic, however, JavaScript is much more useful than HTML by itself. The following JavaScript code displays the date and time in a Web browser:

```
<script>
document.write(Date());
</script>
```

Date() is another JavaScript function that reads the date and time from the clock on the computer where the Web browser is running, and then writes the information on the screen. In this case, the write() function is **calling** or **invoking** the Date function. The Date function does its job, and passes the information back to the function that called it. The write() function then takes the information and displays it as it would any text.

To display the date in a Web page:

1. Use a text editor such as Notepad to copy the code above (for displaying the date and time) into a text file. You can reformat the spacing and indenting of the code, but match the case of the JavaScript exactly.
2. Save the text file as **js-test.html** on your desktop.
3. Open the page in a Web browser to see the current date and time.

Creating Simple Rollovers

Some JavaScript rollovers require using separate functions. The simplest rollovers, however, can be written entirely inline. All JavaScript rollovers involve using JavaScript attributes with A tags. These attributes describe different browser states, similar to the anchor tag attributes in CSS. The most common browser state used for rollovers is the Over state.

In JavaScript, the onMouseOver attribute instructs the browser to do something, such as display a rollover effect, when the pointer moves over the link containing the attribute. It is similar to the anchor tag's HOVER attribute in CSS. The onMouseOver attribute is the core of any JavaScript rollover effect. In the following steps, you will set the onMouseOver attribute to a value which is itself a command. The command tells the browser to display certain text in the status bar at the bottom of the browser window.

To display text in the status bar when users point to a link:

1. Open a text editor such as Notepad and type the following JavaScript code:

   Roll Over Me

JavaScript rollovers require using links. However, if the rollover effect is not intended to be used for a link, you can set the URL to "#".

2. Save the text in a file named **js-test2.html**.
3. Open **js-test2.html** in a Web browser.
4. Point to the link that reads **Roll Over Me** and look to see Hello World! in the status bar, as shown in Figure 10-14.

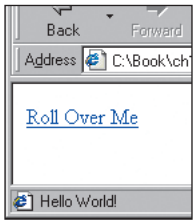

Figure 10-14 Using JavaScript to display a message in the status bar

JavaScript includes additional attributes that describe other possible states:

- **onMouseOut** instructs the browser to do something when the pointer moves off the link. You could have the status bar display one message when the Web page loads, a second message when the mouse is over the link, and a third message when the mouse leaves. You also can use onMouseOut to return a rollover effect to its initial state when the user no longer points to a

link or button. If you do not set this attribute, a rollover effect stays in place once it occurs. For example, if you create a rollover effect that highlights a button when the user points to it, unless you set the onMouseOut attribute, the button remains highlighted even when the user points to another area of the page. With CSS, the HOVER attribute has an implicit definition to return the link to the initial state when the mouse is pulled away. In JavaScript, you must explicitly define this with the onMouseOut attribute.

- **onMouseDown** instructs the browser to perform a task when the mouse clicks the link. The task can be anything from displaying text, to changing color, to opening a new browser window. This is similar to the ALINK attribute in HTML or the ACTIVE anchor tag attribute in CSS.
- **onMouseUp** instructs the browser to perform a task when the mouse button is released after clicking. Again, the task can be anything. This attribute is usually used to return rollover effects to their initial state after being clicked. Links return to the onMouseOver state by default, so use onMouseUp only when you want to display different images.
- **onClick** is similar to onMouseDown, but the effect of this attribute remains in place until the user activates a different rollover state on the page.

In fox.html, the onMouseOver attribute calls a function named changeImages, which swaps the initial image with a second image. The onMouseOut attribute calls the same function to again swap the second image with the initial image.

Creating Graphical Rollover Effects

Recall that you create most rollover effects by swapping one image for another. A basic image swap in JavaScript looks like this:

```
<a href="http://www.course.com"
onMouseOver= "document.image1.src= 'sheep.gif';"
onMouseOut= "document.image1.src= 'dog.gif';">
<img src= "dog.gif" name="image1">
</a>
```

Note the similarities to the anchor tag at the bottom of fox.html.

This JavaScript code initially shows an image of a dog. When the user moves the mouse over the image, the JavaScript replaces the dog image with a sheep image. When the user moves the mouse off the image, the JavaScript replaces the sheep image with the initial dog image.

A key concept is the difference between the document image name and the filename. A filename is associated with exactly one image file, but the file can be displayed multiple times in a Web page. A document filename, such as "image1", is associated with exactly one IMG tag and one place in a Web page, but can display different image files at different times.

The IMG tag is named image1 and the SRC attribute is set to display an image file named dog.gif. The onMouseOver attribute is set to "document.image1.src= 'sheep.gif';". This finds the element in the document with the name image1 (the dog image) and resets the

source to a different image file, named sheep.gif. The onMouseOut attribute is set to "document.image1.src= 'dog.gif';". This finds the same element and resets the source back to the original image. On the Data Disk, open the file named 10-1.html in a Web browser to see the above example. The final effect also is illustrated in Figure 10-15.

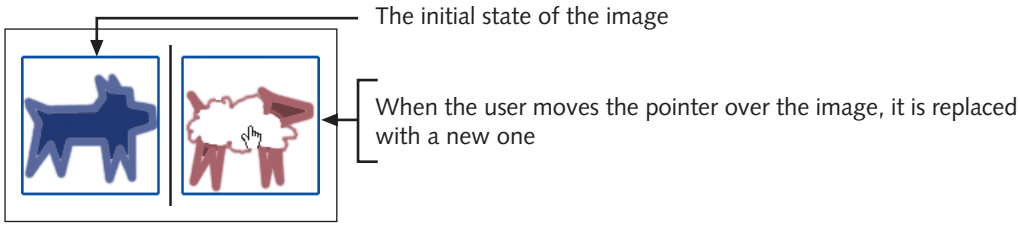

Figure 10-15 A simple rollover effect using onMouseOver and onMouseOut

Creating Multi-Image Graphical Rollover Effects

Every IMG tag on a page can have a name, which then can be controlled by JavaScript. Not only can you create rollover effects to swap the image used as the link, you also can create rollover effects to swap other images on the page.

The following example swaps images for a different IMG tag:

```
<a href="#"
onMouseOver= "document.image2.src= 'sheep.gif';"
onMouseOut= "document.image2.src= 'frog.gif';">
<img src= "dog.gif" name="image1">
</a>
<img src= "frog.gif" name="image2">
```

The onMouseOver and onMouseOut attributes are set to affect the image named image2, which is the frog image. Moving your mouse over the dog image affects the frog image, but leaves the dog image alone, as shown in Figure 10-16. The frog image is not set to invoke a rollover, so nothing happens when you mouse over it.

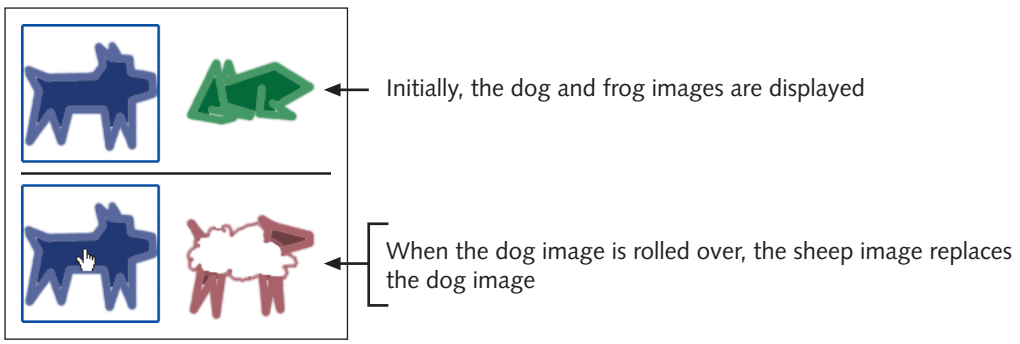

Figure 10-16 A rollover that affects a secondary image

On the Data Disk, open the file named 10-2.html in a Web browser to see the previous example.

You can even set up the JavaScript to swap two images at once:

```
<A HREF="http://www.course.com"
onMouseOver= "document.image1.src= 'sheep.gif';
document.image2.src= 'sheep.gif'"
onMouseOut= "document.image1.src= 'dog.gif';
document.image2.src= 'frog.gif'">
<img src= "dog.gif" name="image1">
</a>
<img src= "frog.gif" name="image2">
```

In the above example, the onMouseOver and onMouseOut attributes both are set to perform two actions, separated by semicolons. Moving the mouse over the dog image sets both image1 and image2 to the sheep image file. Moving the mouse back returns the images to their original states, as shown in Figure 10-17. On the Data Disk, open the file named 10-3.html in a Web browser to see the effect.

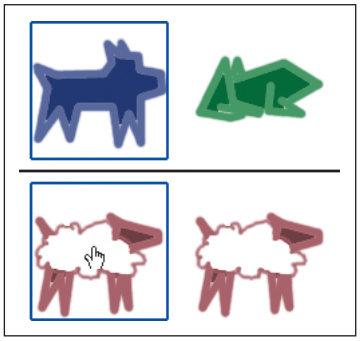

Figure 10-17 A rollover that affects both the activated and the secondary images

Although you can use ImageReady to create rollover effects that swap the image being moused over, you must edit the HTML manually, or use a WYSIWYG HTML editor to create multi-image swaps.

The examples given so far have left out a few elements that are necessary for them to work as well as they can. The following section explains how to optimize rollover effects to make them more efficient.

Optimizing Rollover Effects

You optimize graphics by selecting their format, color depth, and compression level and method. When you optimize JavaScript code, you make the code work more quickly with fewer errors. If you create JavaScript by hand, you can assign different optimization

settings and formats to the images used as different states in rollovers. For example, you might use flat initial-state images and save them as low-color GIFs, and use images with high-color gradations saved as JPEGs for the rollover images. If you use software such as ImageReady to create the rollovers, you must save all states in the same format with the same optimization settings. Then you can edit the JavaScript and again optimize the graphics separately.

The rollover effects used in the past few examples work, but have limitations. Fortunately, you can optimize them to work better. One limitation is that because of the way the JavaScript rollover effects have been written, they must always be written within the anchor tag. If you have many rollover effects, it can be more efficient to write a single JavaScript function to control all rollover effects. The Using Functions section, which follows, explains how to do so.

A second limitation is that the images called by the onMouseOver attribute do not load until the user points to the link. Only then does the browser make a request to the Web server to download the image. If the connection is fast, users might not notice the interruption, but if the connection is slow or if the browser must load many rollover images, users have to wait. To avoid a delay, you can preload the rollover images so that they are already in the cache by the time the user points to a link. The following Preloading Images section explains how to preload images with HTML, JavaScript, and JavaScript functions.

A third limitation is that not all browsers support graphical JavaScript rollovers. Some browsers display the JavaScript code directly in the page, or cause an error when they encounter a rollover. To avoid these problems, you can write the code so that older browsers ignore JavaScript and rollover effects they cannot process. The Making Cross-Browser Rollover Effects section later in this chapter tells you how to hide JavaScript and rollover effects from older browsers and how to specify the script language you want the browser to use.

The remedies for these three limitations are described below:

Using Functions

Instead of writing all of the JavaScript rollover code in the anchor tag, you can write a general-purpose function to swap any image for another. Following is an example of a simple JavaScript rollover function. It can appear anywhere in an HTML page, or can be included in a separate file:

```
<script>
function swapImage(imageName,imageFileName)
{
document[imageName].src = imageFileName;
}
</script>
```

This is a simplified version of the changeImages() function used in fox.html.

The above code defines a new function called swapImage that replaces one image with another. The first line defines the new function and its arguments. The word function indicates that a new function is being defined and its name is swapImage. After naming the function, the code defines the two **arguments** or **parameters** the function uses. The arguments or parameters are always in parentheses and are separated by a comma. Once you define the function name and the arguments it uses, you can use these elements to specify what the function does. The instructions of the function are between the curly braces ({ and }). In this code, the instruction is to take any document image name that is passed to it, and assign to it whatever image filename is passed to it. Thus, a document image, such as "image1", can be assigned to an image filename, such as "frog.gif", and the image of a frog is displayed wherever there is an IMG tag with the name image1.

The following is an example of the swapImage function being called in HTML. Initially, users see the image of a sheep. When users point to the image, it is replaced with the frog image. When they move the pointer away from the link, they see the sheep image again.

```
<a href="http://www.course.com"
onMouseOver="swapImage('image5','frog.gif')"
onMouseOut="swapImage('image5','sheep.gif')">
<img src="sheep.gif" name="image5">
</a>
```


In this example, the onMouseOver attribute calls the swapImage function and passes it two arguments. The first argument is the name of the IMG tag, and the second argument is the image filename that is to be placed in that position. The onMouseOut attribute also calls the swapImage function, setting the source of that image back to the sheep image.

On the Data Disk, open file 10-4.html in a Web browser to see this example.

Functions make multi-image swaps easy. In any anchor tag, include a call to the function, passing the name of the image you want to swap, and the name of the file you want to use.

You can write functions to do almost anything you want. In addition to making the JavaScript more efficient by using less code, you can use functions to improve speed by preloading images.

Preloading Images

Rollover images should swap immediately when the user points to a link. This can happen only if the images are already stored in the browser cache. If they are not, the user has to wait while the images download from the server. You can preload the rollover images to place them in the browser cache before they are needed.

When any Web graphic is downloaded for the first time, it is stored in the Web browser's cache. If the graphic is used again, the browser loads it from the cache instead

of downloading it. Preloading images means downloading them and storing them in the cache without displaying them first. You can preload images using HTML or with JavaScript.

Preloading Images with HTML

The easiest way to preload images is to display them on the Web page, using dimensions so small that the user does not notice them. You can use a normal IMG tag and set the WIDTH and HEIGHT attributes to 1 or even 0. Then the image appears as a small dot. If you use this technique, position the reduced images where they will not be noticed, such as at the bottom of the page.

```
<img src="rollover5.gif" width="1" height="1">
```

Although this method works, it shows dots on the page, which might be an effect you don't want. You also can preload the images using JavaScript. Then the images never appear on the page until they are used in a rollover effect.

Preloading Images with JavaScript

To preload images using JavaScript, first specify which image file the browser should load and with what name it should be associated. Following is an example of JavaScript code to preload images:

```
<script>
var image1=new Image();
image1.src="dog_glow.gif";
var frog=new Image();
frog.src="frog_glow.gif";
</script>
```

This code sets two variables and assigns each of them a particular image file. A **variable** is a placeholder. To create JavaScript rollovers, you use the term "var" to the left of the name of the image you are preloading. The first line declares a variable of the name image1 and defines it as an image by calling the Image() function. The second line takes the image name and sets the source to the image file named dog_glow.gif. When the browser reads this line, it requests the file from the Web server. The third and fourth lines do the same thing, preloading the image file named frog_glow.gif. You can use any variable names you want. In the anchor tag, you can refer to the original image filenames.

When you preload images this way the user does not have to wait for the rollover images to download when the mouse moves over the link. The rollover images are downloaded with all of the other page elements. However, the whole page takes longer to load as it preloads the rollover images. You can use JavaScript functions to preload the rollover images before the images are accessed by a rollover, but only after the rest of the page has finished loading and rendering.

Preloading Images with JavaScript Functions

To preload images with JavaScript functions, your first step is to make a function to preload the images. In the following code, two document images (image1 and frog) are each assigned to an image file (dog_glow.gif and frog_glow.gif):

```
<script>
var image1=new Image();
image1.src="dog_glow.gif";
var frog=new Image();
frog.src="frog_glow.gif";
</script>
```

To convert this code to a function, add the word "function", choose a name, and enclose the instructions in curly braces, as in the following code:

```
<script>
function preLoadImages()
{
     var image1=new Image();
     image1.src="dog_glow.gif";
     var frog=new Image();
     frog.src="frog_glow.gif";
}
</script>
```

This function takes no arguments, as indicated by the empty parentheses after the function name. Your second step is to call this function only after the rest of the page has loaded. To do so, you use a special JavaScript attribute, called onLoad for the BODY tag. This instructs the browser to do something only after all the elements in the body of the Web page have loaded.

Following is an example of a BODY tag calling the preLoadImages function after the other page elements have loaded:

```
<body onLoad="preLoadImages()">
```

Compare the code in the above example with the code in fox.html. Both use preloading functions called by the onLoad attribute in the BODY tag.

You cannot see the advantage of preloading images if you view these examples on your desktop. If you have access to a Web server, try using a graphical rollover effect with and without preloading. The preloaded rollover images appear without making you wait.

In addition to preloading the images, you can optimize your rollovers by making the scripts backward-compatible.

Making Cross-Browser Rollover Effects

There are two ways a browser might have trouble interpreting a JavaScript rollover. An older browser might support some JavaScript features, but does not support setting

sources for images. Alternately, an older browser might not support JavaScript at all, or the user might have turned off JavaScript in the Web browser, disabling it.

If a browser has only limited support of JavaScript, it might not be able to display rollover effects properly. It might display a JavaScript error instead. If a browser does not support JavaScript at all, it will not interpret the SCRIPT tag and will display all the JavaScript code in the page as if it were normal text.

When creating Web pages, you always need to consider just how many users will actually be able to see your work. The percentage of users with JavaScript-incompatible browsers is lower every year as people upgrade to the newest Web browsers. Still, you need to make a few additions to your JavaScript code to make sure that no problems are caused if the JavaScript does not work in a browser. These additions are described in the following sections.

Hiding JavaScript from Older Browsers

To make sure JavaScript code is not displayed in older Web browsers, use comments within the SCRIPT tags just as you do with style sheets. JavaScript comments are the same as CSS comments: two slashes at the beginning of the line.

Following is the sample code that assigns two filenames with two document images, with comment tags added:

```
<script>
<!--
function preLoadImages()
{
   var image1=new Image();
   image1.src="dog_glow.gif";
   var frog=new Image();
   frog.src="frog_glow.gif";
}
//-->
</script>
```

Note how comment tags are used in fox.html. Some comment tags hide text that is intended only for people editing the code, but the comment tags between the SCRIPT tags hide the JavaScript code from noncompliant browsers.

HTML comments frame the code itself, hiding it from the browser. A JavaScript comment to the left of the HTML comment at the bottom prevents a browser that supports JavaScript from interpreting the HTML comment as a JavaScript command.

Another way to prevent the JavaScript from being displayed in older browsers is to place it in the HEAD tag rather than in the BODY tag. Text in the head of a Web page does not appear in the page. If you use comments, it does not matter where you put the JavaScript code, although it is conventional practice to place it at the top of the file.

Specifying Language

JavaScript is not the only language you can use between SCRIPT tags to add functionality to a Web page. You also can use VBScript, which is similar to JavaScript but is based on Visual Basic. To make sure the browser understands which language you are using, you should specify the language using the LANGUAGE attribute of the SCRIPT tag.

Following is the preload script again, with the script language explicitly stated:

```
<script language="JavaScript">
<!--
function preLoadImages()
{
     var image1=new Image();
     image1.src="dog_glow.gif";
     var frog=new Image();
     frog.src="frog_glow.gif";
}
//-->
</script>
```

Most browsers assume the language is JavaScript, but it is safer to state it directly.

Hiding Rollover Effects from Older Browsers

The oldest browsers that support JavaScript support only basic functions, and do not support working with images. You can test whether the browser interpreting the script can work with images. If you include this test in the script itself, it runs whenever a browser reads the script.

The test uses the if statement in JavaScript. This tells the browser that if the condition inside the parentheses is true, continue, otherwise stop. In this case, the condition for which you are testing is whether the browser can work with images in JavaScript.

Following is the preload example with the test for image capability in JavaScript:

```
<script language="JavaScript">
<!--
function preLoadImages()
{
if (document.images)
   {
   var image1=new Image();
   image1.src="dog_glow.gif";
   var frog=new Image();
   frog.src="frog_glow.gif";
   }
}
//-->
</script>
```

After the HTML comment, include the name of the function, followed by an if statement with document.images as the condition. When a browser begins reading the script, it reads the SCRIPT tag and the comments. If it is an older browser, it stops reading until it gets to the end comment tag. If it is a browser with some JavaScript support, it continues reading. When it processes the line "if (document.images)" and its JavaScript support allows it to correctly interpret document.images, it continues through the script, executing all the remaining instructions. If it cannot interpret document.images, it skips the remaining instructions.

In fox.html, each of the three functions tests for this condition.

Most browsers can work with images in JavaScript, so this code is for a minority of users who still use browsers from around 1996.

The JavaScript generated by ImageReady uses functions and preloads the images. Many designers prefer to use the prewritten rollover code from ImageReady or DreamWeaver instead of writing it themselves. The code generated by these programs works across browsers and includes the preloading function described earlier. If you want to create more-sophisticated effects, such as multi-image swaps, generate JavaScript code using ImageReady, and then edit the JavaScript to fit your particular needs.

Chapter Summary

- Rollover effects reinforce users' actions by changing the appearance of images or links when users point to or click them.
- Rollovers can be overused, distracting the user and causing extra downloads. Use rollovers only when they serve a purpose.
- Creating different rollover states in ImageReady is very similar to creating different frames for animations.
- Styles are preconfigured combinations of multiple layer effects.
- You can use layer styles to automate the creation of simple JavaScript rollovers.
- Most rollover effects can be created with tools such as ImageReady. For more complex effects, you must edit the JavaScript code manually.
- Unlike HTML, the JavaScript programming language displays output that depends on its input.
- JavaScript can be written inline or in separate functions. Often you will use both procedures to create rollover effects.
- You should preload the images used in rollover effects so the user does not have to wait for them to download.
- You should hide JavaScript code from browsers that cannot interpret it, and test to make sure that JavaScript-compatible browsers can display the rollover effects.

Review Questions

1. How many images in total are needed if you use Over and Down states with an image?
   a. 1
   b. 2
   c. 3
   d. 4
2. What HTML tag must be used with every rollover effect?
   a. A tag
   b. IMG tag
   c. P tag
   d. SCRIPT tag
3. What states do people use most often for rollover effects?
   a. Click and None
   b. Down and Up
   c. Normal, Over, and Out
   d. Normal and Over
4. Which of the following statements is true?
   a. The Custom state allows you to create additional rollover states.
   b. The names of rollover image files are set in the Name field in the Slice palette.
   c. The None state defines the image shown in the initial state, before the user mouses over it.
   d. You can select multiple states to adjust their layer properties at the same time.
5. Why should you preview a rollover in a browser?
   a. To make sure the frame disposal method works properly
   b. To make sure the images actually swap
   c. To make sure the delay times are accurate
   d. To make sure the looping works properly
6. What changes to a layer do not have to appear in all rollover states?
   a. Applying filters
   b. Applying layer styles
   c. Changing opacity
   d. Changing position

7. In what palette do you choose the names for rollover images?
   a. Animation palette
   b. Layers palette
   c. Rollover palette
   d. Slice palette
8. What files are created in addition to the actual images when you save a rollover in ImageReady?
   a. An HTML file only
   b. An HTML file, a JavaScript file, and a folder named images
   c. An HTML file and a folder named images
   d. An HTML file and a JavaScript file
9. How do you add animation effects to a rollover image?
   a. Create the animation file separately and manually insert it into the HTML file.
   b. Create the animation using the Animation palette, then import it into the Rollover palette.
   c. It cannot be done.
   d. Select the state in the Rollover palette, then create the animation in the Animation palette.
10. Which of the following statements is false?
    a. JavaScript is not the same as VBScript.
    b. JavaScript ignores white space.
    c. Lines of JavaScript code must end in semicolons.
    d. JavaScript must always be written inside SCRIPT tags.
11. Because it includes parentheses at the end of its name, what is rollover()?
    a. an argument
    b. a function
    c. a state
    d. a variable
12. Where would this code be placed in a Web page?

```
onMouseOver="document.a.src='b.gif'; document.c.src= 'd.gif'"
```

    a. between STYLE tags
    b. between SCRIPT tags
    c. between A tags
    d. inside an IMG tag

13. What do we know from looking at the code in question 12?
    a. The number of images swapped by this rollover
    b. What happens when the mouse clicks down on the image
    c. What happens when the mouse rolls off the image
    d. What image is being rolled over
14. What does the following code contain?

```
language="JavaScript"
```

    a. an HTML attribute and value of the SCRIPT tag
    b. an HTML attribute and value of the STYLE tag
    c. a JavaScript attribute and value of the SCRIPT tag
    d. a JavaScript attribute and value of the STYLE tag
15. What is the purpose of the code in question 14?
    a. It tells the browser that the rollover is not CSS-based.
    b. It tells the browser that the rollover is not HTML-based.
    c. It tells the browser that the rollover is not Java-based.
    d. It tells the browser that the rollover is not VBScript-based.
16. What browser will not display JavaScript rollovers?
    a. IE 4
    b. Netscape 3
    c. Netscape 4
    d. Netscape 6
17. What is the difference between Java and JavaScript?
    a. Java does not run in the browser.
    b. JavaScript can be run only in a Web browser.
    c. JavaScript is not a programming language.
    d. They are two names for the same thing.
18. What would this script do?

```
if (document.images) {document.write("Success")}
```

    a. cause an error
    b. do nothing
    c. write the word Success on the Web page
    d. write the word Success on the Web page, only if the browser supports JavaScript rollovers

19. What would this script do?

```
var foo=new Image(); foo.src="bar.jpg";
```

a. It cannot be determined.
b. Preload an image file named bar.jpg
c. Preload an image file named foo
d. Swap an image named foo with one named bar.jpg

20. How can you create rollover effects that will work in every browser?

a. This cannot be done.
b. The JavaScript must be hand-coded.
c. The rollover state images in Photoshop must be edited.
d. The code generated in software such as ImageReady must be used.

Hands-on Projects

Project 1: Analyzing Rollover Effects on the Web

You want to incorporate rollover effects on your Web site, but first you want to see what other designers are doing with them.

Complete the following steps:

1. Using your Web browser, find at least three Web sites that use rollover effects.
2. Move your pointer over the images.
3. Are the images preloaded? You can tell by rolling the pointer over the images, and noticing whether there is a delay. If the rollover images are preloaded, there should be no delay.
4. How many states are used? Do the images use only Normal and Over, or do the effects include additional images for Down and Up?
5. Do any of the sites use rollover effects for instructional purposes, such as swapping in explanatory text messages? Or, are the effects only for navigation buttons?
6. Look at the source code used on the pages. Are the rollover effects created with a JavaScript function? Can you tell what the script does?

Project 2: Using Animation in a Rollover in ImageReady

You want a simple rollover effect, but do not want to do more than change the colors of the initial image. Animate the MouseOver state image so that it rotates when the mouse is over it.

Complete the following steps:

1. In ImageReady open the **house.gif** file from the Data Disk.
2. Select **Duplicate Layer** two times from the Layers palette menu.

3. Select the **top layer**.
4. Select the **Edit** menu, point to **Transform**, then click **Numeric**.
5. In the dialog box, type **10** under Rotate and click **OK**.
6. Select the **middle layer** and rotate it by **-10** degrees.
7. Open the **Rollover palette** and duplicate the **Normal** state. Make sure it is of type Over.
8. Select the **Over** state and open the **Animation palette**.
9. Click the **Duplicate Frame** button at the bottom of the Animation palette.
10. Select the **second frame**, and then deselect the **visibility icons** for the middle and bottom layers.
11. Select the **first frame** and deselect the **visibility icons** for the top and bottom layers.
12. Set the delay for both frames to **0.1** seconds and set the looping to **Forever**.
13. Play the animation to confirm that the house rocks back and forth.
14. Open the **Rollover palette** again.
15. Select the **Normal** state and deselect the **visibility icons** for the top and middle layers.
16. Preview the rollover in a browser. The house should be straight. When you roll over the house it should wiggle back and forth.
17. Optimize and save the rollover images and HTML in a new folder named **project_10-2**. Save the HTML file as **house.html**.

Project 3: Use Styles to Automate Rollover Creation in ImageReady

You want a quick, simple rollover effect. Use layer rollover styles to quickly make a three-state rollover effect.

Complete the following steps:

1. In ImageReady, create a new **36-pixel square image** with a **transparent** background.
2. Use the Paint Can tool to fill the layer with any color.
3. Show the **Styles** palette.
4. To make sure you have the default style set displayed, select **Reset Styles** from the Styles palette menu.
5. Click the style named **3-state Gradient** from the Styles palette.
6. In the Optimize palette, select the preset named **JPEG High**.
7. Show the **Slice** palette.
8. In the Name text box, type **button**.

9. In the URL text box, type **next.html**.
10. Preview the rollover in a browser.
11. In ImageReady, click the **File** menu, click **Save Optimized As**, and save the HTML file as **button.html** in a new folder named **project_10-3**.

Project 4: Using Different Optimization Settings for Rollovers in ImageReady

In ImageReady, all layers and states must be optimized using the same settings. This is inconvenient when the states require different settings. You can create a rollover using weaker optimization, then later you can further optimize specific states.

Complete the following steps:

1. In ImageReady, create a new **image**, **50** pixels high and **100** pixels wide with a **transparent** background.
2. In the Rollover palette, create an **Over** state.
3. In the Layers palette, duplicate the layer.
4. Select the **lower layer**.
5. Use the Paint Can tool to fill the layer with **white**.
6. Select the **Buttons** style set from the Styles palette menu.
7. Click the style named **Clear Embossed**.
8. Select the **upper layer**.
9. Using **red** text, type the word **Vote!** with the Type tool. Center the text.
10. Select the **Over** state and select **Match Layer across States** from the Rollover Palette menu.
11. Select the **Normal state** and make sure only the upper layer is visible.
12. Select the **Over state** and make sure both layers are visible.
13. Open the **Optimized tab** in the image window. In the Optimize palette, adjust the settings to **8-color GIF**.
14. Select each of the **states** to view in the Image window with the Optimize tab selected. The Normal state should look fine, but the Over state will look banded.
15. In the Optimize palette, select the preset named **JPEG High**. Both states should look fine with this setting, but the first state could be optimized much more.
16. Save the optimized files and HTML file in a new folder named **project_10-4**. Name the HTML file **vote.html**.
17. Find the image used for the Normal state, open it in ImageReady and reoptimize it again with **8 colors** as a **GIF**.
18. Save the image as **vote_01.gif**.
19. In a text editor, edit the HTML file to reflect the new name.

Project 5: Create a Secondary-Image Rollover with ImageReady

Edit the HTML from Project 3 to create a rollover effect that affects a secondary image. You cannot create the rollover for this exercise with ImageReady alone. You must edit the rollover's HTML code.

Complete the following steps:

1. Open the file named **button.html** from Project 3 in a text editor.
2. Save the file as **button2.html** in a new folder named **project_10-5**.
3. Copy the **images** folder from project_10-3 to project_10-5.
4. Add the following IMG tag to the bottom of the page, above the </BODY> tag:

```
<IMG NAME="button2" SRC="images/button-down.jpg">
```

5. Edit the values of the attributes of the A tag so that the four calls to the changeImages() function use **"button2"** as a parameter instead of using "button".
6. Edit the line beginning with ONMOUSEUP to use **"button.jpg"** as a parameter instead of using "button-over.jpg".
7. Save the HTML file and open it in a browser. Rolling over and clicking the button on the left should cause the image on the right to swap.

Project 6: Create a Simple Rollover in JavaScript

You are creating another Web site and want the rollover effect to be visible by as many users as possible. However, you need to create the page as quickly as possible. Knowing you can use a more-sophisticated method later, use inline JavaScript to create the rollovers.

Complete the following steps:

1. In Photoshop or ImageReady, create **three round buttons 72** pixels high and **72** pixels wide with **transparent** backgrounds.
2. In **black** text, type the words **one**, **two**, and **three** on the buttons.
3. Optimize and save the images as **one.gif**, **two.gif**, and **three.gif** in a new folder named **project_10-6**.
4. Select the **Paintbrush** tool and set the foreground color to **yellow**.
5. Select each **image** in turn, and add a new layer behind the text. Use a **65-pixel feathered brush** to add a **faded yellow spot** behind the text.
6. Save these new images as **one_glow.gif**, **two_glow.gif**, and **three_glow.gif**.
7. Open a new text file and save it as **rollover.html** in the project folder.

8. Copy the following HTML into the file:

```
<a href="one.html"
       onMouseOver= "document.one.src='one_glow.gif';"
       onMouseOut= "document.one.src='one.gif';">
<img src="one.gif" name="one" border="0"></a>
```

9. Duplicate this code twice, changing every reference of "one" to **"two"** and **"three"**, respectively.
10. Save the file and view it in a browser. Test to make sure the three images change as you mouse over them, and change back as you roll away from them.

Project 7: Create MouseDown Rollovers with JavaScript

You have a next button that is too plain. Make it more interesting by using ImageReady to add a second image for the MouseDown state.

Complete the following steps:

1. In ImageReady, create a **50-pixel square image**.
2. Make the foreground **green** (**#009900**) and select the **Line** tool.
3. Select the **Create filled region** box. Set the Weight to **5**. Select an **arrowhead** at the end of the line. Set the shape to **500**% width and **250**% length.
4. Draw a **line** from left to right across the middle of the Image window. Hold down the **Shift** key to keep the line horizontal.
5. Select **Duplicate Layer** from the Layers palette menu.
6. In the lower layer, turn the arrow **black** by using the Hue/Saturation window and setting the Lightness to **-100**.
7. Select the **Gaussian Blur** filter and blur the layer at a radius of **2** pixels.
8. Open the **Rollover palette** and click the **New** button at the bottom of the Rollover palette to duplicate the state. Change the type of the new state to Down by holding down the **mouse** button over the state name and selecting **Down**.
9. Select the **Normal** state in the Rollover palette. Select the **upper layer** in the Layers palette.
10. Select the **Move** tool. Press the **up arrow** key four times and the **left arrow** key three times. This should move the green arrow up and to the left of the black arrow.
11. Click the **Rollover Preview** button in the toolbox or in the Rollover palette.
12. The arrow should appear to float above its shadow. Click the **image** in the Image Window. The arrow appears to be pressed down.
13. Click the **Preview in Default Browser** button in the toolbox. You should see the arrow and all necessary JavaScript below it, including all preloading functions.
14. Select **Save Optimized As** from the File menu. Save the HTML page as **arrow.html** in a new folder named project_10-7.

Project 8: Create Image Swaps Controlled by Text Links

Rollover effects do not have to involve rolling over images. Create images containing text that describes the links used to display those images.

Complete the following steps:

1. In Photoshop or ImageReady, create three **100-pixel square images**.
2. Use the Type tool to add text messages to each image. In the first, write **The latest news about graphics on the Web**. In the second, write **See sample images in our photo gallery**. In the third, write **Biographies of our staff**.
3. Optimize and save the images as **text1.gif**, **text2.gif**, and **text3.gif** in a new folder named **project_10-8**.
4. Create a **1-pixel transparent image**. Save it as **clear.gif**.
5. Open a new text file and save it as **text.html** in the project folder.
6. Copy the following code into the text file. The swapImage() function needs only one argument, since it will be making changes to only one IMG tag.

```
<script language="JavaScript">
<!--
function swapImage(imageFileName)
{
       document.textImage.src = imageFileName;
}
//-->
</script>
<img src="clear.gif" name="textImage" height="100" width="
100">
<br>
<a href="news.html"
 onMouseOver="swapImage('text1.gif')"
 onMouseOut="swapImage('clear.gif')">News</a>
```

7. Duplicate the anchor tag twice and edit the link text to read **Gallery** and **Who's Who**.
8. Edit the name of the image file used for the MouseOver states to **text2.gif** and **text3.gif**.
9. Save the HTML file and open it in a browser. Rolling over the three links displays the relevant text image above each link. Rolling off the links returns the image to the blank, transparent image.

Case Project

By this point, your portfolio should almost be finished. It should contain pages for your gallery and pages for your resume and biography. The pages should be linked together using a navigation bar with buttons. Now you should add rollover effects to the buttons in the navigation bar.

- You created a set of navigation buttons for the chapter on buttons. Create duplicates of these to use as rollover graphics. The rollover versions should change color in a way that highlights the text and distinguishes them from the nonrollover buttons.
- Use JavaScript from ImageReady or from one of the examples in this chapter to create the effect.
- In addition to the buttons, create 150-pixel square graphics with text descriptions of the different sections in your site.
- When the user rolls over a navigation button, the rollover image should appear, and the related text image should also appear on the side.